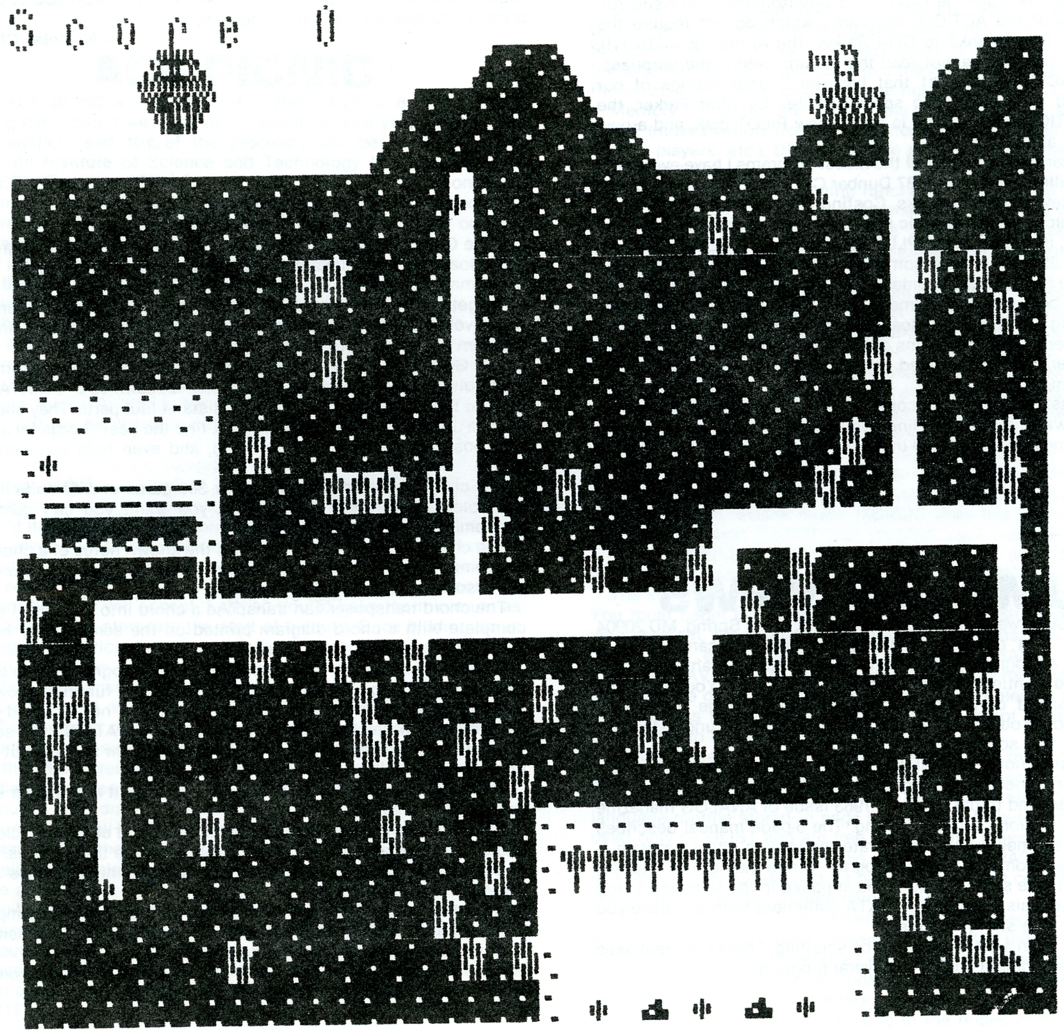
ACE
Atari Computer Enthusiasts
3662 Vine Maple Dr. Eugene OR 97405

APRIL, 1984
Mike Dunn & Jim Bumpas, Editors



Sydney Brown: DIAMOND MINE

News and Reviews

by Mike Dunn, Co-Editor

There is not a lot of news from the Atari Company yet, but our President, Kirk Stockwell is going to the San Francisco Computer Faire and should come back with lots of new info — see next month's ACE. However, several ex-Atari employees are coming out with "Super-Atari-like" computers. The first, mentioned several months ago, is, according to reviews, a super graphic IBM-PC compatible 80186 computer from **MindSet** (617 N. Mary, Sunnyvale, CA 94086). Designed by a team of ex-Atari people headed by Roger Badertscher, the head of the Atari Computer division in the early years, it has several special super-graphic chips like the GTIA, etc. It is supposed to have graphics and animations superior to what you see on TV cartoons, all for under $3000 for a complete unit. It is so good many of the software houses now producing top Atari software such as Synapse and DataSoft have already produced some fantastic games and graphic utilities for it. Looks like a real winner and I will love to get my hands on one.

Another ex-Atari designer, Jay Minor, who designed the Atari super chips now works for **Amiga** and has designed another fantasitic graphics/animation machine costing less than the IBMPCJr and be compatible with it.

Another new program annouced for the Atari is **Jane**. This is an integrated package with a word processor, Calc and database programs and uses a mouse. It is supposed to be like a combination of the Apple Mac programs and Lotus 123. By Arktronics (113 S. 4th Ave., Ann Arbor, MI 48104) I am looking foward to seeing it.

Closer to home, we have several new disks ready for the exchange library, including a **"Best of ACE #8"** for $15 including an Iron-on ACE patch. This disk includes the TinyText update from the Dec. issue, run time versions of the ACTION! programs which do not require the ACTION! cartridge (thanks to Clint Parker, the Author of ACTION!), KONG! for those who cannot read the listings, and some surprises. Another is **ACTION! Disk #1** that has the Source listings of our programs and more, including several utilites by Clint Parker, the author of ACTION! ($15). There is also a new PILOT disk, and a new Utility disk is almost ready.

For a real bargain and some of the nicest programs I have ever seen, the **Jacksonville ACE** group (1187 Dunbar CT., Orange Park, FL 32073) has sent us several of their disks. Costing only $8 each, they include some of the nicest music/graphic programs I've seen/heard, and some great utilities, including MENUPLUS, which is on our BEST of ACE #8 above. Disks 25 and 29 have some great programs, and their Music disk has to be heard. Please contact them for details. We charge more for our disks because the income is used to support our Newsletter (no advertising!) and Bulletin Board. Speaking of the Bulletin Board, there have been no new programs put on it recently because of lack of room, however, we have ordered two 8"drives which should give us plenty of room.

Last months listings did not come out well. The ACTION! listing by Stan Ockers was mixed up (no line numbers to check!) and the KONG listing was smeared. Elsewhere in this newsletter are the corrections. Sorry!

BUMPAS REVIEWS

Scroll I ($20, Superware, 2028 Kingshouse Rd., Silver Spring, MD 20904 301-236-4459) is a machine language program you can use as a subroutine in a program of your own design. Your programs may now easily include continuous scrolling in 8 directions. You may also control the speed of the scroll, and you can change the ANTIC character mode. You can do all this without having to understand all the details of fine scrolling.

The manual is not a tutorial, but use of this program can be very instructive for someone learning about scrolling. The user is referred to "De Re Atari"and to the October, 1983 issue of ANALOG magazine for more information on fine scrolling. The 5-page manual describes the simple steps needed to incorporate the routine into a program.

The disk also contains a demo program with a map of the Western Pacific Ocean. The scrolling routine is supplied in the form of a string variable, but it is also supplied in DATA statement format in case you cannot use a long string variable.

I've always been a bit intimidated by scrolling. This package makes me want to experiment with it in several programs.

BRUCE LEE

In **BRUCE LEE** (Datasoft, 19808 Nordhoff Pl., Chatsworth, CA 91311, $30), Datasoft points the way to the next level of arcade games. As Bruce Lee, you must work your way through 20 rooms to find and destroy the evil wizard. Each room in the wizard's oriental castle has a series of hanging lanterns which, if knocked out singly or in a combination, will open a secret passage to the next room.

Sound simple? If might be, except while you're doing this, you are being pursued by a large sumo wrestler called a Green Yamu, and a black Ninja. These, along with the Sword, Flaming Bushes, and energy traps create a myriad of hazards.

You use the joystick to make Bruce Lee climb, run, leap, duck, chop and kick. Points are scored by entering new rooms, striking and knocking out the Ninja and the Yamu, and knocking down lanterns. Bruce Lee starts with 5 "falls" and earns an additional one for every 40,000 points.

The graphics are supurb, and the challenges quite varied. The manual implies that if you should win and destroy the wizard, the next sortie into the wizard's castle will be tougher. I've only been able to survive through the first 10 rooms, so I haven't been able to check this point out yet.

Besides the basic 1 player vs. computer, Datasoft has provided two other options: 2 players taking turns vs. computer, and player vs. player. This last option is unique and will give this game a longer life than most arcade-style games. One player runs Bruce Lee while the other runs the Green Yamu. This, in effect, gives the computer as much intelligence as your opponent. All in all, I find **BRUCE LEE** to be a most challenging and satisfying program.

— Nickolas Chrones

GUITARADE

The Guitarist's Home Companion $29.95 by Richard Lindgren
(Enabler Software, P.O.Box 58, Lamoni, IA 50140)

To get away from the usual reviews about wargames, adventures, etc., I volunteered for this review. As an avid guitar player, I find this program especially interesting.

The Guitarist's Home Companion is a utility to help you improve your guitar playing. I must say the sound capabilities of the ATARI are used to the fullest. This program consists of four parts. They allow the user to display over 300 guitar chords, find the best chords for a song, transpose chords into different keys, and even help you tune your guitar.

The chord encyclopedia holds over 300 guitar chords and can play the notes which make them up. If you want, you can dump the diagrams to a printer.

The chord finder has the user input the known notes of a chord and the computer will find and play suitable notes to fill out the chord. You can also dump this to printer.

The chord transposer can transpose a chord into any key. It comes complete with a chord diagram printed on the screen. Very helpful. Once again you can dump to a printer.

I feel the guitar tuner is the best part of the program. I have used it to tune my bass guitar and a six-string guitar. A function allows the user to move up or down octaves or play notes not included in the main menu. I have never seen the sounds of the ATARI used so well. There was some distortion at very high and very low sounds, but this is negligible.

There is another function to change the default mode from sharps to flats.

Overall, I this is a very good job all around. The chord encyclopedia could be larger, around 1000 chords, but it is fine the way it is. When combined with a printer and a stereo hookup, this becomes a very powerful tool.

Useful for all levels of guitar players. This program is compatible with ATARI 800, 800XL, 1200XL with a disk drive. The documentation is quite complete and easily understood.

— Aaron Ness

Boulder Dash

(First Star Software, $30)

This is a very good arcade-style game modeled after the popular Dig-Dug game. The fact that it's produced by First Star, a company well known for it's hi-res graphic pages, should give you some idea of the well-done screens.

In the game you move a constantly digging ant-like creature called Rockford through the earth, avoiding death by a falling boulder, or by any of the cave's other occupants. To score points, you must collect rainbow colored jewels which dot the different caves, while keeping an eye on the ever falling time limit.

After all the jewels in a cave are collected, you advance to the next cave, which, of course, is more difficult. In the beginning caves, there are just boulders and jewels, but beyond these 'easy' levels are ones containing deadly fireflies, butterflies, or even the dreaded Amoeba. These seldom visited lower levels are extremely difficult because of combinations of these characters.

There are options for one or two players, a pause feature, and an allowance for difficulty selection. The documentation is adequate, but could be better.

The game has its funny parts too. For instance, if you pause for a bit, Rockford will begin tapping a foot, patiently waiting for a response from the player.

This is a popular game around our house. Even my parents enjoy playing it, and that's pretty rare for them to enjoy an arcade style game. The best attribute of the game is challenge. This is one of the hardest games I've yet seen for the Atari. If you love a good challenge, this one's for you.

— Tim Ebling

ACE PICNIC

The ACE picnic will be at Jasper Park, August 8, 1984. We're planning early, aren't we? One of the events at this year's picnic will be an auction, and 10% of the proceeds will be donated to the Willamette Institute of Science and Technology (known locally as "WISTEC"). The other 90% of the proceeds will go to the persons who put items up for auction. The items must be computer related, and software must be original retail copies with the complete original documentation.

This will be an opportunity for people to exchange their software which they no longer use. New users can try out some commercial software for a low price. People wanting to upgrade hardware might find a buyer for old hardware. Those who cannot attend this year's picnic may still participate in the auction. Sellers must ship their items to Bumpas & Ridder, Attorneys at Law, 492 East 13th Ave., Eugene, OR 97401. Items which are not accompanied by enough money to reship to the seller and which remain unsold after the auction will not be returned until the seller arranges to pay for the shipping.

Sellers must notify Jim Bumpas of the name and description of any items they wish to sell before 1:00 pm, Friday, June 22, 1984. The reason for this is the July newsletter will contain a notice announcing the list of items for sale at the auction. The list of auction items will be sent to anyone accompanying their request with a self-addressed and stamped envelope. This will give those out of town time to review the list to see if they wish to bid on any items. Sellers may specify a minimum bid they will accept. I advise you to keep the minimum as low as possible to stimulate the start of bidding. Once bidding on an item begins, the price rises dramatically. We won't spend a lot of time on each item. If no one bids promptly, we'll pass the item and it'll probably remain unsold.

Buyers from out of town will list the items for which they want to bid and the highest price they will bid. These lists will not be seen by any bidders at the auction until it's over. Auction officials will bid $1 higher than the current bid on the selected items until the seller's limit is reached. Absentee bidders must accompany their lists with a check made out to ACE drawn in an amount equal to at least 25% of the maximum total of all bids they wish to make. Buyers whose checks do not clear before the day of the auction will not have their bids executed. If any balance is owing for purchased items, buyers must pay the amount due within 10 days after the auction, or else these items will be sold to the next bidder on the item. Buyers must pay for all shipping costs. Buyers attending the auction must pay cash for items purchased.

Sellers will receive cash or check for 90% of the price their items bring at auction, plus a receipt for the other 10% signed by an official of WISTEC.

XBASIC

(by George Schwenk (c) 1983 by SUPERware 2028 Kingshouse RD, Silver Spring MD 20904)

Description: Atari BASIC enhancement
Medium: Disk or cassette
Requirements: disk drive or cassette recorder, Atari BASIC, 16K RAM (XBASIC uses less than 3K of RAM)
Cost: $29.95

XBASIC adds a library of thirty new functions which will help you accomplish otherwise time-consuming programming chores with relative ease. Each of these machine language subroutines is accessed by the USR command from Atari BASIC.

The disk version of XBASIC contains an initialization file and six examples of short programs which use several of the functions. A thirty-page manual concisely describes each of the functions with examples of how to incorporate them in BASIC programs, but does not explain much about how it all works. However, SUPERware offers to provide the source code to those who might wish to modify or better understand what Mr. Schwenk has done. The cost for the source code is $5. Furthermore, permission is given to include XBASIC in commercial software, provided credit is given to SUPERware.

An example of one of the functions in XBASIC is DLIST. By typing X = USR(DLIST), you will get a disk directory listing without calling DOS.

If you have wished Atari BASIC had string arrays, you might appreciate XBASIC's four string functions. SDIM dimensions string arrays; PSTR puts a character string into a string array; GSTR gets a character string from an array; INSTR searches a string array for a substring. Also included are three similar array functions for integers.

I/O functions are PCHRS, GCHRS, SSAVE and SLOAD. Use PCHRS to put bytes from RAM to a disk or cassette file; GCHRS gets them back. These are great for saving or loading your own character sets. SSAVE saves the screen; SLOAD loads it from disk or cassette. SMOVE, a related function, changes the RAM area to be displayed on the screen, which is useful for page flipping.

Memory functions are CLM (clear memory), FILL (fill memory), MOVE (move memory from one location to another; used for character sets or players, etc.), DPEEK (double peek a two-byte sequence) and DPOKE (a two-byte poke).

Also included are two timing functions. STIMER (set timer) sets a countdown timer. DELAY waits for the amount of time you specify.

Two sound functions, VDIM and XSOUND, are used to play a sequence of sounds during vertical blank interrupt, one sound each 1/60th of a second.

XBASIC has a special graphics call (SGR) which makes it easy to set up ANTIC modes 4 and E, with or without a four-line text screen. For example, "GR.0:DUM = USR(SGR,2)"sets up a full-screen four-color ANTIC mode 4 screen.

Finally, XBASIC provides seven player/missile graphics functions, which enable, move, set the size and color of, and detect collisions of players and missiles. PMOVE moves the desired player to any x,y location and uses shape data from any part of RAM.

There are two ways to use XBASIC. The first is to add it to an existing program. The procedure is to LIST "D:YOURPROG" (or LIST "C:"), LOAD XBASIC, and then initialize XBASIC by typing GOSUB 32500 in the immediate mode. [XBASIC uses line numbers 1-12 and 32490-32502.] Next, ENTER your program back into memory. Finally, type X = USR(XSAVE):SAVE "D:YOURPROG" (or CSAVE). This procedure will save XBASIC along with your BASIC program.

The second method is to use XBASIC when writing a new program. You must begin by loading and initializing XBASIC before typing in your program. When editing your XBASIC programs, it is necessary to begin each editing session by initializing XBASIC, and to use X = USR(XSAVE) before saving. I learned that the hard way, although I had read the directions prior to starting. My problem is I like to save my programs frequently, and I occasionally forgot to use XSAVE, or else forgot to initialize XBASIC at start-up.

XBASIC uses page 6. I spent several hours trying to convert a program from a magazine to XBASIC, only to discover it would not operate correctly because it already had a non-relocatable machine language routine stored in page 6. Although 40 bytes (1561-1600) are used only by DLIST and are free for your use unless you use DLIST, you must face the reality of this limitation to XBASIC if you already use other routines designed for page 6. The object code for XBASIC is stored as part of the variable name table.

I find it difficult to recommend XBASIC to beginners, but more experienced BASIC program developers may find these new functions facilitate faster programming, as well as faster program execution. Mr. Schwenk appears to be a conscientious programmer who offers others a way around re-inventing the wheel. And if you are one of those hackers who like re-inventing in the hopes of making a better wheel, perhaps you should take advantage of his gracious offer to provide the source code.

—Deloy Graham

TIDBITS

Tips for Disk Users

This month I am going to start in with some quips of special interest to those with disk drives. But before I do this, I want to address some comments to those of you who enjoyed the first Tidbits articles. Now that I have the attention of all three of you, I will continue. If you yearn to learn of more locations to PEEK and POKE, you should check out the following publications which I feel are very good: **Mapping the Atari** (by COMPUTE! Books), **Master Memory Map** (by Educational Software), and **De Re Atari** (available from the Atari Program Exchange — this is not really a memory map, but it will tell you alot about how to use some of Atari's fancier features). Even the back of the Atari BASIC reference manual contains a short list of locations you may wish to POKE around.

First, a POKE of interest to all users. Location 65 controls whether or not you will hear the buzzing and beeping of disk and cassette input/output operations. POKE it with a 0 to turn the sounds off, a 1 to turn them back on again. This may be useful in a professional applications program or if the noise irritates you, but personally, I like to be able to hear what is going on.

If you ever want the files in your disk directory to be in a certain spot or order, this next tip is for you. First of all, place a freshly initialized disk in your drive and write the following dummy program:

```
10 DIM D$(20):FOR A=1 TO 10
20 D$="D:JUNK":D$(7)=STR$(A)
30 SAVE D$:NEXT A
```

Now to put a program wherever you want, simply delete the dummy program in the spot you wish to use, and SAVE out your chosen program. The reason this works is that DOS always uses the first empty spot it finds in the directory. You can change the values in the FOR-NEXT loop if you want more or less 'slots' reserved.

I have some friends with Apple computers who are always giving me a hard time about my 800 and its disk operating system. They enjoy making claims about how easy it is for them and how hard it is for us to manipulate disk files from BASIC. Well, they don't know about the XIO command. It allows Atari BASIC to lock, unlock, rename, and delete files as slick as the proverbial whistle. It's so easy I won't even explain it here — page 30 of the BASIC reference manual gives examples, and, if those aren't clear enough, just look over the TinyDOS listing from a couple of months ago.

Have you ever wondered what is in your Atari's memory? Some day if you have nothing to do and want to try something different, type the following immediate-mode line:

```
FOR A=600 TO 32000:?CHR$(PEEK(A));:NEXT A
```

Watch for words to appear. Most of what you see is of absolutely no meaning to you, but you may notice something interesting. Now, if we change the 600 to a 40960 and the 32000 to a 49152, we can see the inside of the BASIC cartridge. If you are patient enough, you're bound to come across some BASIC words. So what? Well, if you can PEEK memory, you should be able to POKE it, too, so we can change words, etc., to what we want. Unfortunately, the BASIC cartridge is ROM, which means we can POKE it all we want and it will just ignore us. But we can change RAM, and so we could change such things as DOS. Program 1 this month is just such a program. It is for those of you with OS/A+ 2.10 who might like to change the commands. When run, this program will show you the existing command, and ask you for your command. Hitting only RETURN will result in the command not being changed, otherwise, enter in your three letter command and press RETURN. The program will POKE your changes into memory. If you wish to make the commands part of the system whenever you boot up, use the INIT command to write a new DOS file to your disk.

— Dale Lutz
Canada

TECHNICALLY SPEAKING

(reprinted from the Dec., 1983 issue of the Huntsville, Alabama Atari Users Group Newsletter)

Last month I showed an example of a machine language routine called from BASIC via the USR function. That routine to porform a search was binary loaded on page 6 of memory. There are several different ways to incorporate these routines into a progam . . .

The method I want to demonstrate involves converting the machine code into an ATASCII string. The BASIC program has to dimension a string variable and then define the variable. The machine language routine is then available with a USR call to the address of the string variable. Since the string is defined by the program, it is saved as part of the program and does not have to be poked into memory. Another advantage is you don't have to limit yourself to a routine which must fit into a safe part of memory (i.e., page 6) because BASIC will keep track of the routine as a string, preserving it. Since we do not know where in memory the string will be at a given time, it is generally best to make the routine relocatable. The search routine I showed you last month is relocatable; all branches were relative branches without JMP and JSR instructions. Therefore the routine could be anywhere in RAM and work properly. If you must have your routine in a certain place in memory, you can have the routine relocate itself to that location. Of course you will have to move it to a location you have protected.

Converting a machine language routine to an ATASCII string could be a lot of work. Fortunately I wrote a BASIC program to do all the work for you. Listing 1 is a program which will read the binary file (your object code for the routine) from the disk and generate the BASIC lines to dimension and define the string as ML$. You then input a filename and the program LISTs those lines to the disk. Since this program generates those lines beginning at line number 1, your BASIC program should start with a line number of 10 or greater depending on the length of the machine language string. Then you only have to load your BASIC program, ENTER the file with the lines for the string, and resave the program.

Since BASIC will not allow you to define a string variable with a quotation mark character and since an EOL (return) does not have a printable screen character, line 1130 of Listing 1 checks for these codes and changes them to question marks. The program then generates additional lines for the BASIC program which defines those individual codes with the CHR$ function. I took the search routine from last month, ran this program on it, and merged the generated lines with last month's BASIC test program. Listing 2 is the result. Line number 1 dimensions ML$ and line 2 defines the string variable as the ATASCII representation of the machine code. If there had been a code in the routine which represented an ATASCII quotation mark, then a third line would have been generated reading:

```
3 ML$(x,x)=CHR$(34)
```

where x is the position of the "in the string. The only change necessary to this program is at line 80 where USR(1536,...) is changed to USR(ADR(ML$),...). If you typed in the assembly source code last month, give these two programs a try to see how easy this method is. If you use the program at Listing 1 to convert your own ML routine to ATASCII strings, keep in mind it will work only with object codes saved as normal Atari DOS simple binary files. This means you must save your code in one step using no append function.

— C. Mueller

****
M.U.L.E. TOURNAMENT
12 NOON, APRIL 15, 1984
2160 West 11th Avenue, Eugene
$3 Entry Fee
All fees = Prizes for Winners
for more information, call:
Jim Bumpas 484-9925
****

Shane Rolin: PROTECT YOUR PROGRAMS

This article will describe to the user several ways of protecting 'SAVEd' Atari Basic programs. The main problem in protecting Basic programs is with the 'SAVE' to disk or cassete command. The task of protecting a 'SAVEd' programs is difficult because of the many different ways a program can be stopped by the user or by errors. I will explain the following methods:

1: Protecting against the [BREAK] key.
2: Protecting against [SYSTEM RESET].
3: 'BLOWING' the Variable Name Table.
4: Erasing End of Statement pointers.
5: Other.
6: Routines to do the above.

Protecting against the [BREAK] key

The Break key has always presented problems for protection because when the user presses the [BREAK] key, program execution is stopped, and the program is then listable. Disabling takes two simple POKE values. The two values are:

```
POKE 16,112
POKE 53774,112
```

POKE 16,112 is a POKEY interrupt. It's shadow is 53774. POKE 112 into both of these locations in order to prevent the [BREAK] key from stopping your program. It should be noted that the GRAPHICS command will re-enable the [BREAK] key, so the user should POKE these values very frequently. A good idea is to use the Machine Language Subroutine that I have written. Put the subroutine at line 999 and in your program define the variable 'BREAK' to equal 999 and throught your program execute several 'GOSUB BREAK' commands. The subroutine will execute and then RETURN.

Protecting against [SYSTEM RESET].

When the user presses the [SYSTEM RESET] key, a JMP $A04D is executed in the BASIC cartridge which is the Warmstart subroutine, and all previous 'POKEd' values are returned to default. The two POKE values to disable the [SYSTEM RESET] key are:

```
POKE 580,1
    or
POKE 9,255
```

Location 580 is the coldstart flag. If this location is 'POKEd' with a 1 (powerup in progress flag is set) , the computer will reboot whenever the [SYSTEM RESET] key is pressed. Any non-zero number between 1 and 255 will also do the same.

Location 9 is the boot flag success indicator. When 'POKEd' with 255, the user program will lockup when [SYSTEM RESET] is pressed.

Use the Machine Language routine that I have already described. It also contains these two POKE values.

'Blowing' the Variable Name Table.

Blowing the VNT is one method of preventing a program from being modified. This will allow the program to be listable, but all of the variable names will be 'messed' up. By executing the second routine that I have written, it will change all of the variable names to a RETURN, which will make the program almost inpossible to understand. Execute this routine by a GOTO 32700 command from direct mode.

Erasing End of Statement Pointers.

Erasing End of Statement Pointers is a good way to protect programs from being 'LOADed', 'LISTed', and modified. By executing the third subroutine I have written through a GOTO 32700 command, the current statement pointer will be set to 0 and will allow your program to 'RUN' normally even though it cannot be 'LISTed' or 'LOADed'.

Other methods.

Another method of program protection is by doing both Blowing the VNT and Erasing the ESP's. The fourth routine I have supplied does just that. Execute this routine by doing a GOTO 32700 from direct mode. This will almost make the program impossible to change through any other method except the program that I have written to reverse this process which will soon appear in the newsletter.

ROUTINES

```
0 REM Listing 1:
1 REM Disable the [BREAK] and
2 REM [SYSTEM RESET] keys.
3 REM
4 REM Machine Language Subroutine
5 REM by Shane Rolin
6 REM
7 REM TO USE:
8 REM  Simply put 'X=USR(1536)'
9 REM  in any program line.
10 REM
100 FOR A=0 TO 17:READ B
110 POKE 1536+A,B:NEXT A
120 DATA 104,169,112,133,16
130 DATA 141,14,210,169,1,141
135 DATA 68,2,169,255,133,9,96
140 REM
150 REM Subroutine in M/L as follows:
160 REM PLA
170 REM LDA #$70
180 REM STA $10
190 REM STA $D20E
200 REM LDA #$01
210 REM STA $0244
220 REM LDA #$FF
230 REM STA $09
240 REM RTS
250 REM
```

```
0 REM Listing 2:
1 REM Subroutine to change all
2 REM variables to CHR$(155)-
3 REM the RETURN character. The
4 REM user may change the POKE
5 REM value in line to any ATASCII
6 REM character.  Change the SAVE
7 REM in line line 32710 to your
8 REM own filename or C:
9 REM
10 REM by Shane Rolin
11 REM
12 REM TO USE:
13 REM Type GOTO 32700 in direct
14 REM mode.
15 REM
16 REM
32700 FOR VNT=PEEK(130)+PEEK(131)
      *256 TO PEEK(132)+PEEK(133)*256
32705 POKE VNT,155:NEXT VNT
32710 SAVE "D:filename"
```

```
0 REM Listing 3:
1 REM Subroutine to erase the
2 REM End of Statement Pointers.
3 REM
4 REM by Shane Rolin
5 REM
6 REM TO USE:
7 REM Type GOTO 32700 in direct
8 REM mode.
9 REM
32700 POKE PEEK(138)+PEEK(139)
      *256+2,0:SAVE "D:filename":NEW
```

```
0 REM Listing 4:
1 REM Subroutine to change all
2 REM variables to CHR$(155)-
3 REM the RETURN character. The
4 REM user may change the POKE
5 REM value in line to any ATASCII
6 REM character.  Change the SAVE
7 REM in line line 32710 to your
8 REM own filename or C:
9 REM   This program also erases
10 REM  the End of Statement
11 REM  Pointers.
12 REM by Shane Rolin
13 REM
14 REM TO USE:
15 REM Type GOTO 32700 in direct
16 REM mode.
17 REM
100 FOR A=0 TO 17:READ B
110 POKE 1536+A,B:NEXT A
120 DATA 104,169,112,133,16
130 DATA 141,14,210,169,1,141
135 DATA 68,2,169,255,133,9,96
140 REM
150 REM Subroutine in M/L as follows:
160 REM PLA
170 REM LDA #$70
180 REM STA $10
190 REM STA $D20E
200 REM LDA #$01
210 REM STA $0244
220 REM LDA #$FF
230 REM STA $09
240 REM RTS
250 REM
999 X=USR(1536)
32700 FOR VNT=PEEK(130)+PEEK(131)
      *256 TO PEEK(132)+PEEK(133)*256
32705 POKE VNT,155:NEXT VNT
32710 SAVE "D:filename":NEW
```

MEETING

Weds Apr 11

South Eugene High Cafeteria

7:30 PM Hear about the Faire

David Fuller: GLOBAL CHANGE

```
1 REM ********************************
2 REM ** GLOBAL FILEMANAGER CHANGES **
3 REM **        by David Fuller     **
4 REM * reprinted from H.A.C.K.    **
5 REM *(Helpful Atari Computer     **
6 REM * Knewsletter, Feb 9, 1984   **
7 REM ******************************
10 TRAP 1020:GOSUB 1100:PRINT CHR$(125
);TTL$
20 REM ** GET FILE NAME
30 PRINT :PRINT " Insert Filemanager D
ata Disk"
50 PRINT :PRINT " Enter name of file "
;:INPUT ANSR$
60 FILE$=ANSR$:FILE$(LEN(FILE$)+1)=".F
MT":FLNAM$=ANSR$:IDX=1
70 REM ** GET AND DISPLAY FIELDS
80 PRINT CHR$(125);TTL$:POKE 752,0
90 OPEN #1,4,0,FILE$
100 INPUT #1;NUMFLDS
110 INPUT #1;FLDN$
120 FOR N=1 TO 5:INPUT #1;A:FN(N)=A:NE
XT N:A=0
130 CLOSE #1:LE=LEN(FILE$):FILE$(LE-3,
LE)=".IDX":OPEN #1,4,0,FILE$
140 FOR N=1 TO 5:INPUT #1;IDX1:NEXT N:
INPUT #1;IDX2:INPUT #1;IDX3
150 PRINT :PRINT "File ";CHR$(34);FLNA
M$;CHR$(34);" has ";NUMFLDS;" fields"
160 PRINT :PRINT ,"Field names"
170 PRINT LIN$:A=0:B=5:C=3
180 FOR N=1 TO NUMFLDS*12 STEP 12:A=A+
1
190 B=B1:IF A=11 THEN C=20:B=6
200 IF A=10 THEN C=2
210 POSITION C,B:PRINT A;". ";FLDN$(N,
N+11)
220 NEXT N:POSITION 2,16:PRINT LIN$
230 IF IDX=0 THEN PRINT " NO INDEX FOU
ND ":GOTO 280:REM "(INVERSE)"
240 PRINT "Indexed fields:  ";
250 IF IDX1>0 THEN PRINT IDX1;
260 IF IDX2>0 THEN PRINT ",";IDX2;
270 IF IDX3>0 THEN PRINT ",";IDX3;
280 PRINT :PRINT :PRINT " Which field
to search on ";:INPUT SRCHFLD
290 PRINT "Which field to change ";:IN
PUT CHGFLD
300 REM ** DISPLAY INSTRUCTIONS
310 PRINT CHR$(125);TTL$:CHGFLD$=FLDN$
((CHGFLD-1)*12+1,(CHGFLD-1)*12+12)
320 SRCHFLD$=FLDN$((SRCHFLD-1)*12+1,(S
RCHFLD-1)*12+12)
330 PRINT :PRINT "* ";CHGFLD$;" has ";
FN(CHGFLD);" characters."
340 PRINT "* ";SRCHFLD$;" has ";FN(SRC
HFLD);" characters.":PRINT
350 PRINT LIN$:PRINT " If entry is enc
losed in Quotes"
360 PRINT "          EXAMPLE:";CHR$(34);"
TEST";CHR$(34)
370 ? " any record in which this field
 ":? "contains the letters between":?
" the quotation marks will be changed.
"
380 PRINT LIN$:PRINT " If entry is enc
losed in asterisks":PRINT "       EXAMP
LE: *TEST*":PRINT " any records in whi
ch this field
390 PRINT " not contain the letters be
tween":PRINT " Asterisks will be chang
ed.":PRINT LIN$
400 REM ** GET CRITERIA
405 CHG$=" ":CHG$(255)=CHG$:CHG$(2)=CH
G$
410 POSITION 2,18:PRINT "Search for: "
;:INPUT CRIT$
420 POSITION 2,19:PRINT "Change to: ";
:INPUT ANSR$
425 CHG$(1,FN(CHGFLD))=ANSR$
430 POSITION 2,20:PRINT "Use Query (Y/
N): ";:INPUT ANSR$
440 IF ANSR$(1,1)="Y" THEN QUERY=1:GOT
O 460
450 QUERY=0
460 POKE 752,1:POKE 764,255:POSITION 5
,22:PRINT " CORRECT (Y/N/End)?"
470 PE=PEEK(764):IF PE=255 THEN 470
480 IF PE=35 THEN POSITION 2,18:PRINT
"XXXXX";:POKE 764,255:POKE 752,0:GOTO
410
490 IF PE=42 THEN RUN :REM "E"
500 IF PE=43 THEN 520:REM "Y"
510 GOTO 470
520 REM ** BEGIN SEARCH
530 PRINT CHR$(125);TTL$:RN=1
540 FILE$(1,1+LEN(FLNAM$))=FLNAM$:FILE
$(LEN(FILE$)+1)=".DAT"
550 CLOSE #1:OPEN #1,12,0,FILE$
560 X=1:FOR N=1 TO NUMFLDS
570 IF SRCHFLD=N THEN STSRCH=X:LNGTHSR
CH=FN(N)
580 IF CHGFLD=N THEN STCHG=X:LNGTHCHG=
FN(N)
590 X=X+FN(N):NEXT N
600 REM ** GET FILE AND CHECK FOR MATC
H
610 FOR N=1 TO LEN(CRIT$):X$=CRIT$(N,N
)
620 IF X$="*" THEN STYPE$="<>":POP :GO
TO 650
630 IF X$=CHR$(34) THEN STYPE$="=":POP
 :GOTO 650
640 NEXT N:STYPE$="=>":GOTO 690
650 LNTH=LEN(CRIT$):IF CRIT$(LNTH,LNTH
)<>"*" OR CRIT$(LNTH,LNTH)<>CHR$(34) T
HEN 670
660 CRIT$=CRIT$(2,LEN(CRIT$)):GOTO 680
670 CRIT$=CRIT$(2,LNTH-1)
680 LNTH=LEN(CRIT$)
690 NOTE #1,SEC,BYTE:INPUT #1;INPT$
700 POSITION 2,2:PRINT "Record #: ";RN
710 POSITION 2,4:PRINT SRCHFLD$;":";
720 POSITION 5,6:FLD1$=INPT$(STSRCH,ST
SRCH+(LNGTHSRCH-1)):PRINT FLD1$
730 POSITION 2,13:PRINT CHGFLD$;":";
740 POSITION 5,15:FLD2$=INPT$(STCHG,ST
CHG+(LNGTHCHG-1)):PRINT FLD2$
750 POSITION 2,19:PRINT "Changed to: "
;
760 POSITION 5,20:PRINT "XXXXXXXX"
770 IF STYPE$="<>" THEN 830
780 IF STYPE$="=" THEN 880
790 REM ** CHECK FOR = OR >
800 IF FLD1$(1,LEN(CRIT$))=CRIT$ THEN

810 RN=RN+1:GOTO 690
820 REM ** CHECK FOR <>
830 REM
840 FOR N=1 TO LEN(FLD1$)-LNTH
850 IF FLD1$(N,N+LNTH-1)=CRIT$ THEN RN
=RN+1:GOTO 690
860 NEXT N:GOTO 920
870 REM ** CHECK FOR =
880 FOR N=1 TO LEN(FLD1$)-LNTH
890 IF FLD1$(N,N+LNTH-1)=CRIT$ THEN 92
0
900 NEXT N:RN=RN+1:GOTO 690
910 REM ** CHANGE FILE AND PUT BACK
920 IF QUERY=0 THEN 980
930 POKE 764,255:POSITION 5,22:PRINT "
Change this record (Y/N) ?"
940 PE=PEEK(764):IF PE=255 THEN 940
950 IF PE=43 THEN 980:REM "Y"
960 IF PE=35 THEN RN=RN+1:GOTO 690:REM
 "N"
970 GOTO 940
980 POSITION 5,21:PRINT CHG$(1,LNGTHCH
G):INPT$(STCHG,STCHG+(LNGTHCHG-1))=CHG
$
990 POINT #1,SEC,BYTE:PRINT #1;INPT$
1000 RN=RN+1:GOTO 690
1010 REM ** CHECK FOR ERRORS
1020 TRAP 1020:ERR=PEEK(195):ERLIN=PEE
K(186)+256*PEEK(187)
1030 IF ERR=136 THEN 1060
1040 IF ERR=170 AND ERLIN=130 THEN IDX
=0:GOTO 150
1050 POSITION 2,22:PRINT "Error ";ERR;
" at line ";ERLIN:STOP
1060 CLOSE #1:POSITION 10,22:PRINT " D
ELETING INDEX ":FILE$(LE-3,LE)=".IDX":
XIO 33,#1,0,0,FILE$
1070 IF (CHGFLD=IDX1) OR (CHGFLD=IDX2)
 OR (CHGFLD=IDX3) THEN DEL=1
1080 IF DEL=1 THEN POSITION 10,22:PRIN
T " DELETING INDEX ":FILE$(LE-3,LE)=".
IDX":XIO 33,#1,0,0,FILE$
```

Stan Ockers: LIFE

```
;          ACTION! LISTING
; ACE NEWSLETTER, 3662 VINE MAPLE DR
;    EUGENE, OR 97405  $12 YEAR
; LIFE BY STAN OCKERS, APRIL 1984

MODULE ; LIFE in Action!
BYTE x,y,tot,consol=53279,hue
BYTE ARRAY old(400),new(400),near(7)
CARD cnt,pos,pop,gen

PROC Nbors() ; Find # neighbors of sq.
  tot=0 pos=y*20+x
  SetBlock(near,8,0)
  IF y>0 THEN
    IF x>0 THEN
      IF old(pos-21)>0 THEN
         tot==+1 near(0)=old(pos-21)
      FI FI
    IF old(pos-20)>0 THEN
       tot==+1 near(1)=old(pos-20) FI
    IF x<19 THEN
      IF old(pos-19)>0 THEN
         tot==+1 near(2)=old(pos-19)
      FI FI FI
  IF x<19 THEN
    IF old(pos+1)>0 THEN
       tot==+1 near(3)=old(pos+1) FI FI
  IF y<19 THEN
    IF x<19 THEN
      IF old(pos+21)>0 THEN
         tot==+1 near(4)=old(pos+21)
      FI FI
    IF old(pos+20)>0 THEN
       tot==+1 near(5)=old(pos+20) FI
    IF x>0 THEN
      IF old(pos+19)>0 THEN
         tot==+1 near(6)=old(pos+19)
      FI FI FI
  IF x>0 THEN
    IF old(pos-1)>0 THEN
       tot==+1 near(7)=old(pos-1) FI FI
RETURN

PROC Init()
CARD dl
 Graphics(1) x=10 y=10 hue=1
 Setblock(old,400,0) Setblock(new,400,0)
 dl=Peek(560) dl==+256*Peek(561)
 Poke(dl+25,70) Poke(dl+29,6) Poke(82,0)
 Poke(dl+30,6) Poke(656,0) Poke(657,7)
 Print("lIFe") Poke(657,29) Poke(752,1)
 Print("Written in Action!") Poke(656,1)
 Poke(657,22) Print("generation")
 Poke(656,2)
 Poke(657,2) Print("population")
RETURN

PROC Dly(CARD wait)
DO wait==-1 UNTIL wait=0 OD
RETURN

PROC Cursor()
BYTE stk,under
Dly(1000) Position(x,y) PutD(6,120)
Dly(1000) Position(x,y)
under=old(y*20+x)
IF under=0 THEN PutD(6,32)
ELSEIF under=1 THEN PutD(6,42)
ELSEIF under=2 THEN PutD(6,10)
ELSEIF under=3 THEN PutD(6,170)
ELSEIF under=4 THEN PutD(6,138) FI
stk=Stick(0) stk==!15
IF stk&1 THEN IF y>0 THEN y==-1 FI FI
IF stk&2 THEN IF y<19 THEN y==+1 FI FI
IF stk&4 THEN IF x>0 THEN x==-1 FI FI
IF stk&8 THEN IF x<19 THEN x==+1 FI FI
IF STrig(0)=0 THEN
  IF old(y*20+x)=0 THEN old(y*20+x)=hue
  ELSE old(y*20+x)=0 FI Poke(77,0)
  DO UNTIL Strig(0)=1 OD FI
DO UNTIL STrig(0)=1 OD
IF consol=5 THEN hue==+1
  IF hue=5 THEN hue=1 FI FI
IF consol<>6 THEN
DO UNTIL (consol&7)=7 OD FI
RETURN

PROC Fillscr()
Position(0,0) pop=0
FOR cnt=0 TO 399
DO IF old(cnt)=1 THEN PutD(6,42)
   ELSEIF old(cnt)=2 THEN PutD(6,10)
   ELSEIF old(cnt)=3 THEN PutD(6,170)
   ELSEIF old(cnt)=4 THEN PutD(6,138)
   ELSE PutD(6,32) FI
   IF old(cnt)>0 THEN pop==+1 FI OD
Poke(656,1) Poke(657,35) PrintC(gen)
Poke(656,2) Poke(657,15) PrintC(pop)
Print("   ")
RETURN

PROC Nxtgen()
BYTE try,j
FOR y=0 TO 19
DO FOR x=0 TO 19
  DO Nbors() new(pos)=0
  IF old(pos)>0 THEN
    IF(tot=2)OR(tot=3)THEN
      new(pos)=old(pos)
    FI FI
  IF old(pos)=0 THEN
    IF tot=3 THEN
      FOR cnt=0 TO 7
      DO try=near(cnt)
        FOR j=cnt+1 TO 7
        DO IF(near(j)=try)AND try>0
           THEN hue=try EXIT FI OD OD
     new(pos)=hue FI FI
  OD OD
RETURN

PROC Main()
DO
Init() gen=0
DO Cursor() UNTIL consol=6 OD
DO Nxtgen() Fillscr()
   FOR cnt=0 TO 399
   DO old(cnt)=new(cnt) OD
   gen==+1
IF consol=5 THEN EXIT FI OD OD
RETURN
```

GLOBAL (con't)

```
1090 CLOSE #1:FILE$(LE-3,LE)=".FMT":PO
KE 764,255:GOTO 80
1100 REM **  SET UP VARIABLES
1110 PRINT CHR$(125);:DIM TTL$(30):TTL
$="       GLOBAL CHANGE PROGRAM "
1120 DIM ANSR$(255),FILE$(17),FLDN$(24
0),LIN$(40),CHGFLD$(12),FNCT$(40),FLNA
M$(17),CRIT$(30),SRCHFLD$(12)
1130 DIM CHG$(255),INPT$(255),X$(1),ST
YPE$(2),FLD1$(255),FLD2$(255),FN(20)
1140 LIN$="------------------------------
----------":PRINT LIN$:A=0:B=5:C=2
1150 RETURN
```

MEETING

Weds Apr 11

South Eugene High Cafeteria

7:30 PM Hear about the Faire

Sydney Brown: DIAMOND MINE

```
1 REM ********************************
2 REM **ATARI COMPUTER ENTHUSIASTS**
3 REM **      3662 VINE MAPLE DR      **
4 REM **      EUGENE OR  97405        **
5 REM **          $12 YEAR            **
6 REM **                              **
7 REM **        DIAMOND MINE          **
8 REM **             by               **
9 REM **        Sydney Brown          **
10 REM *                              **
100 J=32:Q=0:U=1:HS=Q:LEV=U:GOSUB 3200
0
190 SC=Q:L=U
195 H=3:V=4:EN=Q:DIG=Q:ROK=Q:CY=Q:E=Q:
UP=Q:AH=34:AV=2:COLOR 78:PLOT AH,AV:L=
U:CN=1:I=0:QT=27-LEV
200 ST=STICK(Q):IF ST=14 AND V>4 THEN
LOCATE H,V-U,ZZ:IF ZZ<78 THEN COLOR J:
PLOT H,V:V=V-U
201 IF ST=13 AND V<23 AND L<3 THEN LOC
ATE H,V+U,ZZ:IF ZZ<78 THEN COLOR J:PLO
T H,V:V=V+U
202 IF ST=11 AND H>Q THEN LOCATE H-U,V
,ZZ:IF ZZ<78 THEN COLOR J:PLOT H,V:H=H
-U
203 IF ST=7 AND H<38 THEN LOCATE H+U,V
,ZZ:IF ZZ<78 THEN COLOR J:PLOT H,V:H=H
+U
204 SS=STICK(U):IF PL=1 THEN 210
205 IF SS=14 AND AV>2 THEN LOCATE AH,A
V-U,K:IF K=J THEN COLOR J:PLOT AH,AV:A
V=AV-1:COLOR 78:PLOT AH,AV
206 IF SS=13 AND AV<23 THEN'LOCATE AH,
AV+U,K:IF K=J THEN COLOR J:PLOT AH,AV:
AV=AV+1:COLOR 78:PLOT AH,AV
207 IF SS=11 AND AH>10 THEN LOCATE AH-
1,AV,K:IF K=J THEN COLOR J:PLOT AH,AV:
AH=AH-1:COLOR 78:PLOT AH,AV
208 IF SS=7 AND AH<38 THEN LOCATE AH+1
,AV,K:IF K=J THEN COLOR J:PLOT AH,AV:A
H=AH+1:COLOR 78:PLOT AH,AV
209 IF K=79 THEN 900
210 IF ZZ=80 THEN GOSUB 360
211 IF H=6 AND V=10 AND CY=Q THEN L=3
212 IF (H=21 OR H=31) AND V=21 AND CY=
Q THEN L=2:CY=Q:E=Q
213 COLOR 79:PLOT H,V:IF EN=U THEN GOS
UB 350
214 IF H=3 AND V=4 AND UP=U THEN 600
220 IF I=0 AND CN>QT THEN SOUND U,21,8
,15:I=1:P=SCR+110:FOR W=15 TO 0 STEP -
1:SOUND U,21,8,W:NEXT W:SOUND U,0,8,8
225 CN=CN+1:IF I=0 THEN 280
230 IF I=1 THEN POKE P,0:P=P-1:F=PEEK(
P):POKE P,43:IF F<34 THEN 250
235 SOUND U,77,8,15:POKE P,PEEK(P-40):
POKE P-40,PEEK(P-80):POKE P-80,0
240 FOR W=15 TO 0 STEP -2:SOUND U,77,8
,W:NEXT W:CN=1:I=0:P=SCR+110:SOUND U,Q
+U:FOR W=U TO 35:NEXT W:COLOR J:PLOT H
,V+U:GOTO 900
600 COLOR 32:PLOT 3,4:COLOR 79:PLOT 3,
3
605 SOUND U,Q,Q,Q:FOR W=250 TO 30 STEP
 -10:FOR WW=W TO W-25 STEP -1:SOUND Q,
WW,10,10:NEXT WW:NEXT W
610 POSITION 2,1:? SH$;:POSITION 2,3:?
 "   ";:FOR W=30 TO 5 STEP -1:SOUND Q,
W,10,10:NEXT W
620 POSITION 2,1:? SH$(10,12):POSITION
 2,2:? "   ";:FOR W=30 TO 5 STEP -1:SO
UND Q,W,10,10:NEXT W
630 POSITION 2,1:? "   ";:FOR WW=15 TO
 0 STEP -1:FOR W=30 TO 5 STEP -1:SOUND
 Q,W,10,WW:NEXT W:NEXT WW
650 GOSUB 950
699 LEV=LEV+1:GOTO 949
800 FOR W=1 TO 35:POSITION 2,2:? SH$;:
SOUND U,PEEK(53770),8,15:POSITION 2,2:
? "   ";:POSITION 2,3:? "   ";:NEXT W
810 FOR W=15 TO 0 STEP -0.5:SOUND U,PE
EK(53770),8,W:NEXT W:GOTO 900
899 GOTO 899
900 SOUND U,Q,Q,Q:GOSUB 320:GOSUB 320:
GOSUB 950:K=0
910 POSITION 12,Q:? "6 H F        F I J
"
940 IF PEEK(53279)<>6 THEN POKE 709,IN
T(RND(Q)*15)*16+10:GOTO 940
948 SC=Q:LEV=U
949 POSITION 0,2:? "          ":? "
 ";:GOSUB 32100:GOTO 195
950 POSITION 22,Q:? "LM ";DIG;"    ":FO
R W=U TO DIG:FOR WW=15 TO Q STEP -2:SO
UND Q,7,10,WW:NEXT WW:SC=SC+U:POSITION
 6,Q
955 ? SC:NEXT W:POSITION 22,Q:? " R ";
ROK;"    ":FOR W=U TO 77:NEXT W
960 IF ROK>Q THEN FOR W=U TO ROK:FOR W
W=15 TO Q STEP -U:SOUND Q,21,10,WW:NEX
T WW:SC=SC+2:POSITION 6,Q:? SC:NEXT W
965 POSITION 22,Q:? "         ";:SOUND Q
,Q,Q,Q:RETURN
32000 DIM PR$(50),SH$(13):CB=PEEK(106)
-4:POKE 106,CB:PR$(U,41)="hh[illegible]
[illegible]"
32010 GRAPHICS Q:Z=CB*256:A=USR(ADR(PR
$),Z,4):FOR W=Z+128 TO Z+207:READ D:PO
KE W,D:NEXT W:SH$="VWX ↓←←←←Y@Z "
32015 FOR W=Z+272 TO Z+511:READ D:POKE
 W,D:POKE W,D:NEXT W:POKE 752,U:FOR W=
1536 TO 1551:READ D:POKE W,D:NEXT W
32020 POKE 756,CB:DL=PEEK(560)+256*PEE
K(561):POKE DL+3,69:FOR W=6 TO 28:POKE
 W+DL,4:NEXT W:POKE DL+8,132
32100 POKE 708,54:POSITION Q,Q:? "BCDE
F 0            U                 S";
:POKE Z-11,4:POKE 710,24:POKE 709,14
,Q,Q
250 IF P<SCR+86 THEN POKE P,0:GOTO 800
280 ON L GOTO 290,400,500
290 IF (ZZ=76 OR ZZ=77) AND ST<>15 THE
N GOSUB 300
295 LOCATE H,V-U,ZZ:IF ZZ=82 THEN GOSU
B 310
299 ZZ=255:GOTO 200
300 FOR W=Q TO 15 STEP 2:SOUND Q,35,8,
W:NEXT W:SOUND Q,Q,Q,Q:DIG=DIG+U:RETUR
N
310 EN=U:RH=H:RV=V-U:LOCATE RH,RV-U,XX
:IF XX=82 THEN COLOR J:PLOT RH,RV-U:CO
LOR 82:PLOT RH,RV+U:GOTO 900
319 POKE 77,Q:ROK=ROK+U:RETURN
320 FOR W=30 TO Q STEP -U:SOUND Q,77,6
,W/2:NEXT W:SOUND Q,Q,Q,Q:RETURN
350 LOCATE RH,RV+U,XX:COLOR J:PLOT RH,
RV:RV=RV+U:COLOR 82:PLOT RH,RV:IF XX=7
9 THEN 900
355 IF RV<23 THEN LOCATE RH,RV+U,XX:IF
 XX=J OR XX=79 THEN 350
359 EN=Q:GOTO 320
360 COLOR J:PLOT H,V:LOCATE H-U,V,Z1:I
F Z1=80 THEN H=H-U:GOTO 365
362 LOCATE H+U,V,Z1:IF Z1=80 THEN H=H+
U:GOTO 365
363 LOCATE H,V-U,Z1:IF Z1=80 THEN V=V-
U:GOTO 365
364 V=V+U
365 COLOR 79:PLOT H,V:FOR WW=U TO L:FO
R W=30 TO Q STEP -U:SOUND Q,7,10,W/2:N
EXT W:SC=SC+100:POSITION 6,Q:? SC:NEXT
 WW
366 IF L=3 THEN UP=1
369 RETURN
400 CY=CY+1:IF CY>4 THEN CY=2:R=INT(RN
D(Q)*11)+21:LOCATE R,20,ZZ:E=Q:IF ZZ=8
1 THEN E=U
405 IF CY=2 AND E=U THEN FOR W=14 TO 0
 STEP -2:SOUND 0,7,10,W:NEXT W
410 IF E=U THEN SP=CY+19:COLOR J:PLOT
R,SP-U:LOCATE R,SP,ZZ:COLOR 81:PLOT R,
SP
416 IF CY=4 AND E=U THEN GOSUB 300
450 IF ZZ=79 THEN 900
460 IF (H<21 OR H>31) AND CY=4 THEN L=
U:CY=Q:E=Q
499 ZZ=255:GOTO 299
500 IF CY=Q THEN COLOR 204:PLOT 7,V:R=
INT(RND(Q)*6)+U:COLOR J:PLOT R,9:CY=U
505 IF V=8 THEN COLOR 204:PLOT R,9:L=U
:COLOR 61:FOR W=U TO 6:PLOT W,11:NEXT
W:COLOR J:PLOT 7,10:CY=Q:GOTO 299
510 CY=CY+U:IF CY>10 AND CY<17 THEN CO
LOR J:PLOT 17-CY,11
515 IF 17-CY<H AND V=10 THEN 550
549 GOTO 299
550 COLOR J:PLOT H,V:COLOR 79:PLOT H,V
```

```
32110 POKE 712,128:POSITION 18,U:? "SL
MT          ^_   USML";:POKE 512,0:POKE
 513,6:POKE 54286,192
32120 POSITION 15,2:? "SLMLMLMT
 [\] SLMLM";:POSITION 14,3:? "SLMLMLML
MLT     SLMLMLMLML";:POSITION 6,Q:? SC
;
32130 FOR W=4 TO 22:POSITION Q,W:? "LM
LMLMLMLMLMLMLMLMLMLMLMLMLMLMLMLMLMLMLM
";:NEXT W:COLOR 76:FOR W=Q TO 38
32140 PLOT W,23:NEXT W:POKE Z-U,44:SCR
=PEEK(DL+4)+256*PEEK(DL+5)
32150 COLOR 204:PLOT 20,23:DRAWTO 20,1
9:DRAWTO J,19:DRAWTO J,23:COLOR J:FOR
X=21 TO 31:FOR Y=20 TO 23:PLOT X,Y
32160 NEXT Y:NEXT X:PLOT 20,21:PLOT J,
21:POSITION 21,20:? "QQQQQQQQQQQ";:POS
ITION 21,23:? " P U P U P ";
32170 POSITION Q,9:? "LMLMLMLK";:POSIT
ION Q,10:? "L        ";:POSITION Q,11:?
 "L======K";:POSITION Q,12:? "UUUUUUUK
"
32180 GOSUB 32400:NR=49+7*LEV:GOSUB 32
500
32300 FOR W=11 TO 2 STEP -U:POSITION W
,U:? SH$:FOR WW=U TO 21:SOUND Q,31+WW,
10,6:NEXT WW:NEXT W:POSITION 2,U:? "
  "
32310 POSITION 2,2:? SH$;:FOR WW=U TO
21:SOUND Q,31+WW,10,6:NEXT WW:FOR W=40
 TO 160 STEP 10:FOR WW=U TO 21
32320 SOUND Q,W+WW,10,6:NEXT WW:NEXT W
:SOUND Q,Q,Q,Q:COLOR J:FOR W=U TO 21:N
EXT W:PLOT 3,3:POSITION 3,4:? "@";
32399 RETURN
32400 COLOR J:PLOT 3,20:DRAWTO 3,14:DR
AWTO 27,14:DRAWTO 27,12:DRAWTO 38,12:D
RAWTO 38,22:PLOT 17,4:DRAWTO 17,14
32410 DRAWTO 26,14:DRAWTO 26,17:PLOT 3
1,4:DRAWTO 34,4:DRAWTO 34,12:PLOT 34,3
32420 COLOR 80:PLOT 3,20:PLOT 17,4:PLO
T 31,4:PLOT 26,17:PLOT 38,22:PLOT U,10
:RETURN
32500 COLOR 82:FOR W=U TO NR
32510 X=INT(RND(Q)*39):Y=INT(RND(Q)*19
)+4:LOCATE X,Y,ZZ:IF ZZ<>76 AND ZZ<>77
 THEN 32510
32515 IF X<9 AND Y<12 THEN 32510
32520 PLOT X,Y:NEXT W:RETURN
32699 RETURN
32700 DATA 32,136,136,136,136,136,136,
32,32,160,32,32,32,32,32,168
32701 DATA 32,136,8,8,32,32,128,168,32
,136,8,32,8,8,136,32
32702 DATA 136,136,136,136,168,8,8,8,1
68,128,128,160,8,8,136,32
32703 DATA 32,136,128,160,136,136,136,
32,168,8,8,32,32,128,128,128
32704 DATA 32,136,136,32,136,136,136,3
2,32,136,136,136,40,8,136,32
32710 DATA 40,130,128,40,2,2,130,40,0,
0,8,34,32,32,34,8
32711 DATA 0,0,8,34,34,34,34,8,0,0,8,3
4,34,32,32,32
32712 DATA 0,0,8,34,42,32,34,8,168,32,
32,32,32,32,32,32
32713 DATA 128,128,160,136,136,136,136
,136,0,0,32,136,136,136,136,136
32714 DATA 8,8,40,136,136,136,136,40,0
,0,32,170,170,32,0,0
32720 DATA 255,255,255,207,255,255,255
,243,255,63,255,255,255,243,255,255
32721 DATA 40,150,40,20,85,65,65,65,20
,105,20,20,150,150,130,130
32722 DATA 0,0,0,0,32,168,168,32,170,1
70,42,42,8,8,8,8
32723 DATA 16,84,84,85,69,85,21,20,3,3
,15,15,63,63,255,243
32724 DATA 192,192,48,240,252,252,243,
255,0,0,0,0,12,60,207,255
32730 DATA 2,2,9,10,38,37,170,170,170,
170,255,40,40,255,170,170
32731 DATA 128,128,96,160,152,88,170,1
70,85,107,27,27,5,5,1,1
32732 DATA 85,105,100,100,80,80,64,64,
0,10,42,166,149,166,42,10
32733 DATA 130,170,170,102,85,102,170,
170,0,168,170,102,86,102,170,168
32734 DATA 0,0,128,170,170,128,0,0,0,4
0,150,130,150,130,170,136
32740 DATA 72,173,10,210,9,12,141,23,2
08,169,0,141,26,208,104,64
```

```
1 REM *******************************
2 REM * MACHINE LANGUAGE STRING MAKER*
3 REM *          by C. Mueller       *
4 REM *******************************
5 REM
6 REM

1050 N=0:DIM P(20),CODE(20),A$(15)
1060 PRINT CHR$(125):PRINT "ENTER FILE
SPEC OF OBJECT CODE":PRINT "DON'T FORG
ET THE D:":INPUT A$:IF A$="" THEN 1060
1070 OPEN #1,4,0,A$
1080 GET #1,X:GET #1,X:GET #1,LSB:GET
#1,MSB
1090 ADST=LSB+256*MSB
1100 GET #1,LSB:GET #1,MSB:ADEND=LSB+2
56*MSB
1110 LENML=ADEND-ADST+1:DIM B$(LENML)
1120 FOR I=1 TO LENML:GET #1,X
1130 IF X=34 OR X=155 THEN N=N+1:P(N)=
I:CODE(N)=X:X=63
1140 B$(I,I)=CHR$(X):NEXT I
1150 CLOSE #1
1160 LN=1:POSITION 2,12:PRINT LN;" DIM
 ML$(";LEN(B$);")":GOSUB 1290
1170 A=1:B=90:LN=2
1180 IF LEN(B$)<B+1 THEN B=LEN(B$)
1190 POSITION 2,12:PRINT "↑↑↑↑";:POKE
766,1:PRINT LN;" ML$(";A;",";B;")=";CH
R$(34);B$(A,B);CHR$(34)
1200 GOSUB 1290
1210 A=B+1:IF LEN(B$)>B THEN B=B+90:GO
TO 1180
1220 IF N=0 THEN 1250
1230 FOR I=1 TO N
1240 POSITION 2,12:PRINT "↑↑↑↑";:POKE
766,1:PRINT LN;" ML$(";P(I);",";P(I);"
)=";CHR$(CODE(I)):GOSUB 1290:NEXT I
1250 PRINT CHR$(125):LIST 1,LN:PRINT :
PRINT "THESE ARE THE LINES TO BE ENTER
ED AND ADDED TO YOUR PROGRAM"
1260 PRINT :PRINT "UNDER WHAT FILENAME
 DO YOU WANT IT TO BE LISTED TO DISK?"
:INPUT A$:IF A$="" THEN 1260
1270 IF A$(1,1)<>"D" THEN PRINT :PRINT
 "USE D: PLEASE":GOTO 1260
1280 LIST A$,1,LN:END
1290 POKE 766,0:PRINT "CONT"
1300 POSITION 2,10:PRINT ;:POKE 842,13
1310 STOP
1320 POKE 842,12:LN=LN+1:RETURN
```

New ACTION! Bulletin Board by Clint Parker, Author of ACTION! Utilities, etc 716-235-3394

Ed Slawson: COLORS

```
1 DIM A$(1),CC$(20)
2 GRAPHICS 2:POKE 712,0:POKE 82,1:POKE
 83,38:POKE 752,1:SETCOLOR 2,0,0:POSIT
ION 4,4:? #6;"*  CoLoRS  *"
3 ? "by:Ed Slawson,P.O.Box 36,Warren,M
aine":? "      04572 * Feb.1984":FOR D=
1 TO 4800:NEXT D
4 GRAPHICS 0:POKE 752,1:POKE 710,196:P
OKE 709,0:POKE 712,96
5 ? "  This utility was designed to ev
alu- ate either light or dark colored
text"
6 ? "on any of the 128 colors availabl
e in Graphics 0.  It prints out the co
lor"
7 ? "numbers from 1-15 along with the
Lum- inance levels for each color. The
se"
8 ? "are the same numbers used in the
Set- color format."
9 ?
10 ? "  Also printed on screen are the
 Poke values for location 710 ( backgr
ound"
11 ? "register in Gr.0 ). This method
may beused to change colors the same a
s the"
12 ? "Setcolor Statements.  Simply pok
e the number of the color wanted into
the"
13 ? "location of the register desired
."
14 ?
15 ? "  Once running there is plenty o
f texton screen to evaluate which comb
ina-"
16 ? "tions are the most pleasant to t
he eye,and the most readable."
17 ?
18 ? "       Press Any Key To Continue
"
19 IF PEEK(764)<>255 THEN 21
20 GOTO 19
21 POKE 764,255:? "↰":? "↓"
22 ? "       * *   INSTRUCTIONS   * *"
23 ? "↓↓"
24 ? "   To  PAUSE  program at any poi
nt                    just press ANY KEY
"
25 ? "↓↓"
26 ? "   To  RESTART  program at point
 of                      pause, press th
e"
27 ? "                        START KEY"
28 ? "↓↓"
29 ? "  To  EXIT  program, to End or
Rerun,               press the OPTION KEY
"
30 ? :?
31 ? "       PRESS ANY KEY TO CONTINUE
"
32 IF PEEK(764)<>255 THEN 34
33 GOTO 32
34 POKE 764,255
100 GRAPHICS 2
120 SETCOLOR 2,0,0
140 POSITION 1,3:? #6;"WHAT COLOR LETT
ERS"
160 POSITION 4,4:? #6;"ARE DESIRED"
180 POSITION 3,5:? #6;"bLACK OR WHITE"
200 INPUT A$
220 IF A$="B" THEN 320
240 IF A$="W" THEN 340
260 ? "  PLEASE ANSWER ONLY B OR W"
280 FOR Z=1 TO 500:NEXT Z
300 ? :GOTO 140
320 GRAPHICS 0:SETCOLOR 1,0,0:GOTO 360
340 GRAPHICS 0:SETCOLOR 1,9,8
360 SETCOLOR 2,0,0:SETCOLOR 4,0,0
380 FOR COLOR=0 TO 15
400 READ CC$
410 IF CC$="KIM" THEN 645
420 ? "   COLOR #  ";COLOR;"    ";CC$
440 ?
460 FOR X=1 TO 10:NEXT X
480 FOR LUM=0 TO 14 STEP 2
485 IF PEEK(53279)=3 THEN 20000
490 IF PEEK(764)<>255 THEN GOSUB 21000
500 P=PEEK(710)
520 ? "LUMINANCE   ";LUM;,"POKE 710 ("
;P;")"
540 SETCOLOR 2,COLOR,LUM
560 FOR Y=1 TO 200:NEXT Y
580 NEXT LUM
600 NEXT COLOR
640 DATA GRAY,GOLD,ORANGE,RED-ORANGE,P
INK,PINK-PURPLE,PURPLE-BLUE,BLUE,LIGHT
-BLUE,TURQUOISE,GREEN-BLUE
641 DATA GREEN,YELLOW-GREEN,ORANGE-GRE
EN,LIGHT ORANGE
642 DATA KIM
645 ?
20000 GRAPHICS 0:POKE 752,1:POKE 710,1
02:POKE 709,0:POKE 82,2:POKE 201,9
20002 ? "↰↓↓↓ Use the OPTION or SELECT
 button to":? :? "highlight your choic
e below, then"
20004 ? :? "↓ press the START button."
:FOR ME=0 TO 8:POKE 53279,ME:NEXT ME:G
OSUB 20020:SEL=11
20006 POSITION SEL,SEL:? "RERUN THIS P
ROGRAM"
20008 BUTTON=PEEK(53279):IF BUTTON=7 T
HEN 20008
20010 GOSUB 20028:IF CHOICE=6 THEN 200
22
20012 SEL=SEL+2:IF SEL>15 THEN SEL=11:
GOSUB 20020:GOTO 20006
20014 IF SEL=13 THEN GOSUB 20020:POSIT
ION 11,SEL:? "RETURN TO BASIC":GOTO 20
008
20016 IF SEL=15 THEN GOSUB 20020:POSIT
ION 11,SEL:? "RUN MENU PROGRAM":GOTO 2
0008
20018 GOTO 20008
20020 POSITION 11,11:? "RERUN THIS PRO
GRAM":? :? ,"RETURN TO BASIC":? :? ,"R
UN MENU PROGRAM":RETURN
20022 TRAP 20000:POKE 201,10:IF SEL=15
 THEN ? "↰":? :? ,"LOADING MENU":RUN "
D:MENU":TRAP 32767
20024 IF SEL=13 THEN GRAPHICS 0:? :? "
BASIC":? "IS";:POKE 752,0:TRAP 32767:E
ND
20026 TRAP 32767:RESTORE :GOTO 2
20028 GOSUB 20034
20030 CHOICE=BUTTON:BUTTON=PEEK(53279)
:IF BUTTON<>7 THEN 20030
20032 GOSUB 20034:RETURN
20034 FOR ME=0 TO 8:POKE 53279,ME:NEXT
 ME:RETURN
21000 POKE 767,255
21020 IF PEEK(53279)=6 THEN POKE 767,0
:POKE 764,255:RETURN
21040 GOTO 21020
31000 REM THIS UTILITY WRITTEN BY ED S
LAWSON,FINISHED 11/12/83.CHANGES COLOR
S AND LUMINANCE LEVELS TO EVALUATE W
31999 END
32000 GRAPHICS 0:POKE 82,1:POKE 83,38:
POKE 710,196:POKE 709,0:POKE 712,96:?
"↰":? "    READY WHEN YOU ARE !":REM W
```

Graphics by PrintWiz (Allen Macroware, POB 2205, Redondo Bch, CA 90278, $29)

```
0 REM ******************************
1 REM **    POLYCOPY UTILITY      **
2 REM ** FROM JACKSONVILLE ACE    **
3 REM **        DISK # 29         **
4 REM **  501 CRUSIER LANE        **
5 REM ** ATLANTIC BEACH, FL 32233 **
6 REM ******************************
10 GOTO 375
15 GRAPHICS K0:? "POLYCOPY - ATARI version 2.0"
20 ? "(Space for ";INT(BUFF/125);" sectors)":? :IP=-K1
25 GOSUB 250:IF IP=DSN THEN 40
30 IF IP<DSN THEN GOSUB 355:IF Z=YES THEN 25
35 IF IP<K1 THEN 235
40 MAX=IP-K1:? "Type 'Y' if o.k. ";:GOSUB 335:? :IF Z<>YES THEN 235
45 IP=K0:OP=K0:SPLIT=HI:APND=HI:GOTO 55
50 ? :GOSUB 325
55 ADDR=ADR(Y$):ROOM=BUFF
60 IF IP>MAX THEN IP=IP-K1:GOSUB 145:GOTO 240
65 Y=IP:GOSUB 230:TRAP 80:OPEN #K1,K4,K0,DSN$:TRAP TOFF
70 IF SPLIT<>IP THEN ? "Loading ";DSN$;:GOTO 100
75 TRAP 80:POINT #K1,SEC,BYTE:TRAP TOFF:APND=IP:? "Contin'g ";DSN$;:GOTO 100
80 TRAP TOFF:Z=PEEK(195):CLOSE #K1:IF Z<>170 THEN 110
85 ? :? DSN$;" not found,":? "..do you want to try another disk?";
90 GOSUB 350:? :IF Z=YES THEN ? "Insert new disk";:GOSUB 330:GOTO 65
95 X(IP,K0)=K0:IP=IP+K1:GOTO 60
100 X(IP,K0)=ADDR
105 SIZE=USR(CIO,K1,7,ADDR,ROOM):Z=PEEK(851):IF Z<128 OR Z=136 THEN 115
110 CLOSE #K1:? :GOSUB 370:GOTO 95
115 ? " size=";SIZE;:X(IP,K1)=SIZE:IF Z=136 THEN 125
120 ? "/";:NOTE #K1,SEC,BYTE:SPLIT=IP
125 ? :CLOSE #K1
130 ADDR=ADDR+SIZE:ROOM=ROOM-SIZE:IF ROOM>K0 THEN IP=IP+K1:GOTO 60
135 GOSUB 145:IF SPLIT<>IP THEN IP=IP+K1:IF IP>MAX THEN 60
140 GOTO 50
145 IF ROOM=BUFF THEN RETURN
150 ? :? "Insert 'to' disk";:GOSUB 330
155 ADDR=X(OP,K0):SIZE=X(OP,K1):IF ADDR=K0 THEN 220
160 Y=OP:GOSUB 230:Z=8:IF APND=OP THEN Z=Z+K1
165 TRAP 185:OPEN #K2,Z,K0,DSN$:TRAP TOFF:IF APND=OP THEN ? "Append'g ";:GOTO 175
170 ? "Writing ";
175 ? DSN$;" size=";SIZE
180 Z=USR(CIO,K2,11,ADDR,SIZE)
185 Z=PEEK(867)
190 TRAP 190:CLOSE #K2:TRAP TOFF:IF Z<128 THEN 220
195 IF Z<>162 THEN GOSUB 370:GOTO 220
200 ? "Disk full, try another? (answer first)";:GOSUB 335:IF Z<>YES THEN 235
205 TRAP 210:XIO 33,#K2,K0,K0,DSN$
210 TRAP TOFF:IF APND<>OP THEN ? "Insert new disk";:GOSUB 330:GOTO 155
215 IP=OP:SPLIT=HI:APND=HI:POP :GOTO 50
220 IF OP<IP THEN OP=OP+K1:GOTO 155
225 RETURN
230 DSN$="D:":DSN$(K3)=X$(Y*K12+K1,Y*K12+K12):GOTO 360
235 ? :? "User aborted!! "
240 GOSUB 355:IF Z=YES THEN 15
245 CLR :END
250 GOSUB 325:CLOSE #K3:OPEN #K3,6,K0,"D:*.*":IF IP<K0 THEN IP=K0:X$="":? "C-copy, Q-quit, any key to skip "
255 INPUT #K3,Y$:IF LEN(Y$)<17 THEN 320
260 DSN$="":FOR Z=K3 TO 13:IF Z=11 THEN DSN$(LEN(DSN$)+K1)="."
265 IF Y$(Z,Z)=" " THEN 275
270 DSN$(LEN(DSN$)+K1)=Y$(Z,Z)
275 NEXT Z:IF DSN$="DOS.SYS" THEN 255
280 IF LEN(DSN$)<K12 THEN DSN$(LEN(DSN$)+K1,K12)="            "
285 SEC=K0:TRAP 290:SEC=VAL(Y$(15,17))
290 TRAP TOFF:? IP+K1;:POKE COL,5:? DSN$;:POKE COL,28:? "?";:GOSUB 340
295 IF Z=67 THEN ? CHR$(30);"C":GOTO 310
300 ? CHR$(ERASE);:IF Z=81 THEN 320
305 GOTO 255
310 X$(LEN(X$)+K1)=DSN$:IP=IP+K1:IF IP=DSN THEN 320
315 GOTO 255
320 CLOSE #K3:RETURN
325 ? "Insert 'from' disk";
330 ? ", press any key!";
335 GOSUB 340:? CHR$(ERASE);:RETURN
340 GET #K4,Z:IF Z=27 THEN POP :GOTO 235
345 RETURN
350 GOSUB 340:? :RETURN
355 ? :? "Any more files?";:GOTO 335
360 Z=PEEK(764):POKE 764,HI-K1:IF Z=28 THEN 235
365 RETURN
370 ? "[";Z;"] I/O error on ";DSN$:? "...skipping to next file!":? :RETURN
375 REM Delete lines beginning here!
380 GRAPHICS K0:POSITION 13,12:? "Please wait!";
385 K0=0:K1=1:K2=K1+K1:K3=K2+K1:K4=K2+K2:K12=K4*K3:HI=256:TOFF=40000
390 YES=89:LET ERASE=156:COL=YES-K4:OPEN #K4,K4,K0,"K:"
395 DSN=16:DIM DSN$(16),X$(DSN*K12),X(DSN-K1,K1)
400 CIO=960:FOR Y=K0 TO 42:READ Z:POKE CIO+Y,Z:NEXT Y:POKE 709,PEEK(710)
405 DATA 104,104,104,10,10,10,10,170
410 DATA 104,104,157,66,3,104,157,69
415 DATA 3,104,157,68,3,104,157,73
420 DATA 3,104,157,72,3,32,86,228
425 DATA 189,72,3,133,212,189,73,3
430 DATA 133,213,96
435 MAX=842:APND=35
440 IP=375:OP=450:REM These are the from and to delete numbers!
445 GOTO 460:REM Remove this line to allow deletes!
450 ? CHR$(125):? :FOR Z=IP TO OP STEP 5:? Z:NEXT Z:? "CONT":POSITION K0,K0:POKE MAX,13
455 STOP
460 POKE MAX,K12:BUFF=FRE(K0)-APND:DIM Y$(BUFF):BUFF=BUFF-K3:GOTO 15
```

DEVICE HANDLERS

One of the few things the many books on the Atari don't say too much about are Device Handlers. This may lead you to believe these handlers are very hard to make. They aren't.

The BASIC program with this article is a home brewed Memory Drive which works with BASIC or Assembler. Just enter the program, save it, run it, go to DOS and binary load the file "D:MEMORY.LOD". Go to BASIC and do a ?USR(1536). Then load a BASIC program. To test this handler, do a LIST"M:". BASIC will dump your program into our handler. Now type NEW. Then Enter"m:"and the handler will load the program back into BASIC. The file you listed will survive a RESET, but the handler won't. The program will also survive DOS, but not a copy of the duplicate file.

Now for making your own handler. First initialize the handler, look at lines 170 to 350. The table from 370 to 430 holds vectors pointing to different IO operations. Whenever you access any handler, CIO JSRs through these vectors to do each operation. For example, you do an OPEN command in BASIC. Then BASIC calls CIO, and CIO jumps through the OPEN vector in the vector table for the device.

All OPEN does is prepare the device for IO. CLOSE ends all the operations to the device. READ gets one byte from the device, and returns it in the Accumulator. WRITE sends the value in the Accumulator to the device. I'm not really sure what STATUS and the JMP to initializing do. The Special vector is used for operations which don't use OPEN, CLOSE, READ or WRITE directly. For example, DELETE or RENAME for the disk handler. All errors are sent with the Y register. If the Y register is 1, the operation is successful. You can, of course, OPEN different devices inside your own handler routine. This could lead, for example, to a handler to OPEN the disk and print the directory.

Now for IO operations: GET and PUT are easy, the variable you specify is sent directly to the device accessed. But what about INPUT and PRINT? Well, through trial and error I figured that all PRINT and INPUT are is a series of PUTs and GETs for each character. How do you tell the length of INPUT and PRINT? PRINTs finish when the entire string is sent, but INPUTs are completed when the handler sends an EOF. When your string is sent, just load Y register with 136, or the OS end of file.

The source listing I include with this article has enough comments to clear up most other questions. If have any more, just call (301) 972-8324 or write me: Greg Menke, 22500 Old Hundred Road, Barnesville, MD 20838.

— Greg Menke

HACKER'S VIEW

Atari 800 XL

If you have been considering buying an XL series Atari but haven't because it is so new it might have some serious bug, you have nothing to worry about. The XL has BASIC built in. This version is FAST — almost half again as fast as normal BASIC. The BASIC is located in bank switched memory toggled on boot-up by pressing the OPTION key.

ANTIC graphics modes 4 and 5 are added. The keyclick is piped out to the TV speaker. There is no internal speaker. The function keys are set flush in the case down the right side. The SYSTEM RESET does a chip reset, resetting the 6502 and other component chips. However, a program will still be in memory intact, and the system won't reboot. You can recover from system crashes by hitting SYSTEM RESET without losing your program!

There is a parallel bus connector allowing direct connection to the 6502. This will permit even more expandability than the regular Atari 800. The 800XL has 64k, but only 37k is free in BASIC without DOS. Atari says the 16k (above 48k) can be accessed only through certain programs.

Like the 1200, the new XLs won't work with some software. The fault isn't with Atari, but with the people who made the software. All Atari software should work. The Translator disk for the 1200 (free from Atari, call: 800-538-8543 or 800-672-1404) will remedy many compatibility problems. NOTE: The Translator won't work on the 600XL because it lacks enough memory.

The manual supplied is the pits. If you don't know BASIC, get a good book. The much vaunted HELP function is a useless waste of memory. All its tests can be duplicated in BASIC with a few lines. If you want to use the HELP key in your programs, PEEK to location 732 to detect if HELP has been pressed.

Because of the modified Operating System, some OS routines are in different locations. Here are a few of interest to Assembly language hackers:

$F2B0 - New location for $F6A4
$F2FD - New location for $F6E2
$F223 - Entry point for Self Test
$F24A - New location for $F63E.

The new XL series of Ataris are a good bet, well worth the purchase price. I give them a resounding "A" in performance and looks! If Atari releases a good technical manual and memory map, these computers will be hard to beat.

— Greg Menke

FLASH
HEWLETT-PACKARD WILL DEMO THEIR NEW LOW-PRICED INK-JET PRINTER AT THE APR MEETING!!

Direct from the factory in Corvallis, OR.

Life in Action!

The game of Life is a good test of the speed of a language. In the game, one to four players arrange an initial 'colony' on a grid in some pattern. Each colony is then transformed into a new colony, the next generation, according to two simple rules:

1) Each organism with 2 or 3 neighbors survives to the next generation.

2) Each empty space with exactly 3 neighbors has a 'birth' of a new organism in the next generation.

In this version, the grid is 20 X 20. Consider how much calculation must be done for each generation. Eight adjacent squares must be investigated for each of the 400 grid positions, a total of 3200 locations. The whole grid is examined before any changes are made. Since this version runs at about 58 generations/minute, you can see how programs created with Action! run quite fast.

Use the joystick to plant organisms for the initial population. Pressing the trigger will set or reset the location. 'Select' allows cycling between four available colors. Press 'Start' when you are ready to view the changing generations. Pressing 'Select' will clear the screen for a new initial population.

You may at first have difficulty in creating colonies which can exist for a very long time. If you want proof that simple arrangements can lead to long lasting populations, try the PI arrangement. The PI is made of a square, three unit on a side, with the central bottom organism missing. Gliders are also interesting. On the same three unit square, put in only the bottom, the middle right and central upper organisms. This glider will repeat itself after four generations and gradually slide off the lower right portion of the screen.

I have modified the rules slightly because of the different colors. When rule 2 is obeyed, the birth organism appears only when there are at least two parents of the same color. It then takes on the color of the parents.

— Stan Ockers

GLOBAL CHANGE

Filemanager 800 Utility by David Fuller

(reprinted from the 2/9/84 issue of HACK, Helpful Atari Computer Knewsletter from Warwick, RI)

Filemanager 800 is one of the best all-round data management programs. It has many nice features like subfiles and changing formats. One thing I felt a need for and didn't have was a GLOBAL CHANGE FUNCTION. After thinking about it for some time, I decided to write one.

If you look at the files on a Filemanager data disk, you'll see four files for each one you created. If your file is named "TEST", you'll see "TEST.DAT", "TEST.IDX", "TEST.FMT" and "TEST.CMP". "FMT" files contain the number, names and lengths of the fields you set up. "DAT" files contain the actual records you typed in. The "IDX" files contain the number of records entered, which fields are indexed, how long the indexes are, and the index itself. As a point of interest, if the program does not find an "IDX" file it automatically goes to the RE-INDEX FUNCTION. The "DAT" file is the only one you have to copy to keep a back-up.

My program allows you to make the same change to as many records as you want without having to enter the information each time. Upon RUNning, the program will ask for the name of the file. Next it puts the list of the file's fields on the screen. You are asked which fields you want to use for search criteria and which field you want to change. These may be the same field. You simply enter the number of the field. You will also see which field or fields are indexed. If you change an indexed field, you will have to re-index the file.

The next screen shows a few instructions for entering the search information. You'll next be asked to enter the search and change information. A "Y" to the "Query" prompt and the Atari will show you each record it proposes to change and give you a chance to keep any given record unchanged. The program gives you one last chance to change the criteria you've entered. You may change the search, change and query information if you choose.

The Atari will then begin processing the records. As each record is read from the disk the search and change fields are displayed on the screen. If the search field matches the search criteria, the Atari will change the information in the second field and put the record back on the disk. If you responded "Y" for Query, the Atari will ask if it's ok to change each record. If you enter "N", the current record won't be changed. "Y" will have to be pressed for each record you want to change. The change will be displayed. The record number is shown to let you know how far along the process has gone.

After all the records have been looked at the Atari will return to the list of fields. If the change field was one of the indexed fields, the index will be erased.

Type the program and save it before running it. I also suggest you make a backup copy of the file you want to change (in case you type something wrong).

DIAMONDS

Low on fuel, you land your space craft on an astroid. The Snargs own a Galactic Diamond Mine here, and you need to steal all the diamonds to fuel your craft. As you move with joystick 1, you dig into the astroid. You may move under a rock if you are quick, but two rocks stacked vertically will surely get you. Falling stalactites in the bottom cave and the draw bridge in the top one also provide hazards.

A second player may control the mine guard near the tank. If you are quick enough, you will refuel your craft before the tank destroys your space craft.

— Sydney Brown

Colors

by Ed Slawson, P.O.Box36, Warren,Maine 04572 * Feb.1984

This utility was designed to evaluate either light or dark colored text on any of the 128 colors available in Graphics 0. It prints out the color numbers from 1-15 along with the Luminance levels for each color. These are the same numbers used in the Setcolor format. Also printed on screen are the Poke values for location 710 (background register in Gr.0). This method may be used to change colors the same as the Setcolor Statements. Simply Poke the number of the color wanted into the location of the register desired. Once running there is plenty of text on screen to evaluate which combinations are the most pleasant to the eye,and the most readable.

TYPING AIDS

I typed in Stan Ocker's Action program, Rats' Revenge and I noticed a couple of errors. At the bottom of page 9, 1st column, the Update procedure seems to end without a "RETURN". At the bottom of the 2d column there appear 6 lines of code between two RETURNs with no procedure name. These 6 lines, and the RETURN which follows them are the remainder of the Update procedure.

The 3d column on page 9 contains only two procedures: Openrt and Setbar. The line "PROC Rats()" appears incorrectly inside the Setbar procedure. Delete this line.

KONG

In Line 10050, the C1$ contains the following characters ("C-" = Control characters): C-, C-, one-up-arrow \ H Esc-Tab-Insert 2-down-arrows ! $ d C-F C-, C-,

The C2$ contains the following characters: C-, C-, 1-up-arrow 1-down-arrow C-I Esc-Tab-Insert \ \ 1-right-arrow C-R C-S 0 C-, C-,

L1$ is: C-, C-, C-X C-X C-H 1-right-arrow inv-y C-Y x H inv-N C-B

L2$ is: C-, C-, C-X C-X C-H C-X 8 X inv-C-X C-H C-H C-X

R1$ is: C-, C-, C-X C-X C-P inv-x 1-inv-right arrow inv-C-X 1-left arrow C-R s @

R2$ is: C-, C-, C-X C-X C-P C-X 1-up arrow C-Z C-Y C-P C-P C-X

In Lines 20100-20500, all the [and \ are inv-[and inv-\

DIAMONDS

In Line 32000, PR$(U,41) is: h h inv-C-E inv-O h inv-C-E inv-M h h inv-C-E inv-T inv-) inv-C-, inv-C-E inv-L inv-) inv-C- inv-C-E inv-M inv-" inv-C-A inv-spc inv-C-, inv-1 inv-L inv-C-Q inv-N inv-H inv-P inv-y inf-f inv-M inv-f inv-O inv-h inv-d inv-T inv-P inv-p C-. inv-C-,

Line 32010 SH$ is: V W X spc 1-down-arrow 4-left arrows Y inv-O Z spc

Line 32170: All the letters being printed are inverse.

-- J.B.

PRINTER INTERFACES

So, you are thinking about getting a printer for your ATARI. You have even convinced your wife/ husband/ mother/ father to let you buy one when you discover ATARI charges around $200 for the privilege of connecting it to their computer. Never fear, alternatives are here! In the three years we have each had an ATARI we have owned or used four different printer interfaces. We hope our experiences can help you become aware of some of the advantages and disadvantages of these magic boxes.

Axiom AT-846

This direct connect parallel printer interface is cheaper ($100) than the ATARI 850. It will support only parallel printers. It connects into the serial bus and has an expansion port so you can put it anywhere in your serial device chain. The cable to the printer is 19" long with a Centronics plug. The I/O cable is 29" long and the "unit" is 5.5"x 3.5"x 1". Although not always needed, an external power supply is provided with a 70" line. If your printer provides 250 milliamps of 5 volt power on pin 18 of the Centronics plug you can switch the unit to run on this power and not use the power adapter. The Axiom provides the capability to make four other modifications to the unit. These allow the unit to support a wider range of printer types and user preferences. The Axiom does pass all eight bits to the printer so if your dot matrix printer will support dot graphics, the Axiom will too. We have run a wide variety of software using the Axiom and it has worked perfectly. The Axiom is well engineered and an excellent product.

Ape Face

We cannot say the same for the Ape Face. This interface has no expansion port so it must be the last peripheral in your I/O chain (like the 410 recorder). This is not convenient for us, and may not be for you. The Ape Face will support only a parallel printer and it also has a Centronics plug. The I/O cable is about 36" and the printer cable 19". It draws power from the ATARI which is OK, but the ATARI is not designed to support this extra load. You may run into a power supply problem and we think we have already seen one. A far more serious problem is the Ape Face is not 100% compatible with all the available software. We could not get Ape Face to function with B/GRAPH by Inhome Software, SUSPENDED by Infocom, and GRAPHIC MASTER by Datasoft. It did work with all our BASIC programs and passed all eight bits. The Ape Face sells for $59, but remember, you get what you pay for.

MPP-1100

An innovative product is the Microbits Peripheral Products MPP-1100 ($100). This unit plugs into joystick port #3, so it won't work on the XL computers. It comes with a chip you must install on your operating system board. Although this sounds terrifying to some, it is an easy and well documented procedure. The purpose of this chip is to allocate joystick port #3 to the printer device inside the ATARI. But MPP doesn't stop there. They improved one feature in the operating system and fixed a bug in another. The MPP-1100 allows LPRINTs in FOR-NEXT loops in the dot graphics mode. A 48" cable to the ATARI is provided along with a 12" cable (Centronics plug) to the printer. Our biggest complaint is the Centronics plug fails to lock onto the Epson connector. Although it is powered from the computer we never noticed any power supply problems. The MPP-1100 is a great product if you have joystick port #3 and don't use it.

[Note from the Editor: While the MPP-1100 is still on the market, MPP now manufactures only the MPP-1150. It's also $100 and needs neither a joystick port nor a printer handler. It's a greatly improved product, and a printer buffer is easily added with a memory chip.]

We have seen at least three articles in various publications showing how to make a "Home Brew Interface". It uses two joystick ports and works just fine, BUT you must load in a driver routine to make it work. This severely limits the uses for this kind of interface. Cable lengths are variable and costs are minimal. See ANALOG Magazine #16 for details.

There are many interfaces on the market and we hope this article will help you select the right one for you. Be alert for interfaces which only pass 7 bits (no dot graphics) and interfaces which only work in specific modes (for example, only when the modem is on).

—Lynn Clock
Phil Fitzjarrell

AVOID DOS!

(reprinted from the Dec., 1983 issue of the Huntsville, Alabama Atari Users Group Newsletter)

You can write DOS to a disk directly from BASIC with the direct mode command (i.e., no line numbers): OPEN #1,8,0,"D:DOS.SYS":CLOSE #1. This will open the drive and write DOS.SYS to a formatted disk. DUP.SYS will not be written. You must CLOSE #1 or END, or the file will be written and left open and not show up on a later Directory search! If you keep getting an error on this attempt with no drive action, type END to close all IOCBs and do it again.

The other DOS functions may also be directly accessed (or from within a program) with XIO commands:

```
FORMAT: XIO 254,#1,0,0,"D1:"
UNLOCK: XIO 36,#1,0,0,"D1:filename.ext"
LOCK:   XIO 35,#1,0,0,"D1:filename.ext"
RENAME: XIO 32,#1,0,0,"D1:filename.ext"
DELETE: XIO 33,#1,0,0,"D1:filename.ext"
```

Can you find the others?

DOUBLE YOUR PLEASURE

(reprinted from the Dec., 1983 issue of the Huntsville, Alabama Atari Users Group Newsletter)

Last week I decided to hook up a second monitor to my Atari — a green screen monitor — primarily for word processing. Then, without changing anything, I could still play games since the color monitor and monochrome monitor would be hooked in parallel, displaying the same thing at the same time. Well, I booted in my word processor and started typing out a letter. All of a sudden I couldn't believe my eyes! You see it turned out I made a mistake and hooked up the second monitor in series with my color monitor. Now I was looking at the text of my letter flowing from the green screen to the color monitor. Yes, I was looking at an 80 column display!

Needless to say I was totally awed with this fortuitous discovery. Well, I later put in the Pac-Man cartridge to play a bit. There I was on the color monitor; the green screen was dark. I exited the left-hand door to escape Inky, but I didn't come back in the right-hand door! No, the green screen to the left lit up a second Pac-Man maze, but with a twist. The joystick directions were reversed. When I pushed left I went right. Those clever people at Atari! I later did some experimenting and found by doing a slight circuit modification to put the vertical blank of the two out of phase and putting one monitor on top of the other, I could have 48 lines of text. Just think what I could do with four monitors. Come to think of it, just think of the possibilities of two printers.

— C.M.

FROM THE LIBRARY

The next edition of the library list will be available about April 10. The January printing has been exhausted and is being revised to include the Best of ACE #8, Utility #3, Pilot #3, and Education #6.

In the "For what it's worth department", I will pass along this: Last week, I received an order from one of our members overseas. He mentioned that blank disks cost him about $5.00 each and asked me to use only one side of the disks for his order and leave the other blank. He said he appreciates the opportunity to save a couple of dollars by using both sides of the disk but in his case it was less expensive to buy than single sided. The choice is yours.

In the mail bag: Last month, we published Dice Probability by Ken Waibel of Hawaii. John Kelley of Portland read it and it stirred up memories of some programs he had written. John went through his collection and polished up YATC, and CRAPS, and wrote 2 programs for his grandchildren and great-grandchildren called Dice Arithmetic and Domino Arithmetic. We will be sharing these with you in the coming months and they will be on Education disk #6. Thanks John!

— Ron Ness

TYPESETTING FROM YOUR COMPUTER

ATARI OWNERS: If you have a modem, text editor, and communications program to send ASCII files, you should consider the improved readability and cost savings provided by **TYPESETTING** your program documentation, manuscript, newsletter, or other lengthy text instead of just reproducing it from line printer or daisy-wheel output. Computer typesetting by telephone offers you high quality, space-saving copy that creates the professional image you want! Hundreds of type styles to choose from with 8 styles and 12 sizes "on line." And it's easy to encode your copy with the few typesetting commands you need.

COMPLETE CONFIDENTIALITY GUARANTEED
— Bonded for your protection —
PUBLICATION DESIGN, EDITING, & PRODUCTION

Editing & Design Services Inc.

30 East 13th Avenue Eugene, Oregon 97401
Phone 503/683-2657

Best of ACE books

Volume 1 are bound issues of the ACE Newsletter from the first issue, Oct 80 to June of 1982

Volume 2 covers July 1982 to June 1983

Only $12 each ($2 extra for Airmail). Available only from:

George Suetsugu
45-602 Apuapu St
Kanoehe, HI 96744

SortFinder 1.2
Composite index of Atari related articles from 5 popular computer periodicals from Apr '81 to June '83, including ACE. Only $6 for ACE member from:

Jim Carr, Valley Soft
2660 S.W. DeArmond
Corvallis, OR 97333

Atari Computer Enthusiasts

A.C.E. is an independent, non-profit and tax exempt computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $12 a year for U.S., and $22 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.
Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405.
**

President	Kirt Stockwell 4325 Sean , Eugene, OR 97402 / 503-689-5355
Vice Pres	Larry Gold 1927 McLean Blvd., Eugene, Or 97405 / 503-686-1490
Secretary	Bruce Ebling 1501 River Loop #1, Eugene, OR 97404 / 503-688-6872
Librarian	**Ron and Aaron Ness** **374 Blackfoot, Eugene, Or 97404 (503)689-7106.**
Co-Editor	Mike Dunn 3662 Vine Maple Dr., Eugene, Or 97405 / 503-344-6193
Co-Editor	Jim Bumpas 4405 Dillard Rd., Eugene, Or 97405 / 503-484-9925
E.R.A.C.E. (Education SIG Editor)	Ali Erickson 295 Foxtail Dr., Eugene, Or 97405 / 503-687-1133
E.R.A.C.E. Corresponding Secretary	Robert Browning 90 W. Myoak, Eugene, OR 97404 / (503)689-1513

Send ~~27c~~ 50¢ stamps or coin (~~50c~~ $1.00 overseas) to the Ness' for the new, updated ACE Library List —new in Feb 84!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400 with a TARA keyboard, 2 double-density double sided disk drives with an ATR 8000 interface, 2 double density Percom disk drives, an Epson MX80 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband, 1206 N. Stafford St., Arlington, VA 22201. See the Nov '82 issue for complete details.

ATARI
COMPUTER
ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL